

AL/COM Chemical Engineering programs cover a number of the more complex and tedious calculations demanded of the chemical engineer, simplifying his routine computation tasks, and permitting him to devote a greater fraction of his time to original and productive design work.

All programs are conversational, eliminating input/output problems and program preparation. It is unnecessary for the user to possess any knowledge of the internal workings of the programs or prior computer programming experience to use the library effectively. Where large amounts of data are required, they may be input either directly on the teletype terminal, or from a previously created data file. All references in the input statements, and in the solutions output, are given in terminology familiar to the chemical engineer.

In addition to the programs presented here, others covering design of equipment for such processes as evaporation and humidification are under development.

MC CABB (McCabe-Thiele Calculations)

Simulates a binary distillation. The equilibrium curve, stream percentages of the least volatile component, feed quality and reflux ratio are entered. The number of theoretical stages and the composition at each stage is output.

HEATUP (Nonisothermal Batch Heat Transfer)

Inputs are the physical property characteristics of the fluids, and system parameters. Output is a time-temperature history of the batch heat transfer. Heatup and cooldown may be done. Many different types of transfer, e.g., coils and jacket, are handled.

TRAYS (Ballast Tray Design)

A distillation column ballast tray is designed using the methods outlined in the F. W. Glitsch Handbook # 4900 (Revised). Output is a complete design, including tower diameter and number of flow passes. Tray efficiency may be calculated if desired, using the AIChE Bubble Cap Method.

STRIPS (Stripping Column Calculations)

The compositions of the exit streams of a stripping column are calculated using the Edmister method. The temperature of the streams, plus the L/V ratios throughout the column are also output. The program is equipped to handle condensing steam.

DISTIL (Multicomponent Distillation)

The compositions of the exit streams of a simple distillation column are calculated using the Thiele-Geddes method for a solution, the theta method for mass balance correction, and the Newton-Raphson method for theta-convergence. The temperature and mass profiles throughout the column may be output if desired.

BUBDEW (Bubble and Dew Point Calculations)

The bubble point and dew point of a multicomponent mix are calculated using classic methods described in "Design of Equilibrium Stage Processes", by Buford D. Smith (McGraw-Hill). If water is present in the mix, a secondary dew point is calculated.

Continued on other side

ISTHRM (Isothermal Flash)

The proportion of vapor and liquid of a multicomponent mix at a given temperature is output. A condensing curve may be output, if desired.

ADFLSH (Adiabatic Flash)

Output is the temperature of a multicomponent mix needed to flash off a desired amount of material. The compositions of the vapor and liquid are also output. A condensing curve may be output, if desired.

LIQLIQ (Liquid-liquid Extraction Calculations)

Program uses concentration data to derive activity coefficients using three-suffix Margules equations. Classical methods described in "Design of Equilibrium Stage Processes" by Buford D. Smith (McGraw-Hill) are used to calculate the composition of the exit streams.

ABSORB (Absorbing Column Calculations)

The compositions of the exit streams of an absorbing column are calculated using the Edmister method. The temperatures of the streams and the L/V ratios throughout the column are also output.

TWOPHS (Two-phase and Flashing Flow Pressure Drop Calculations)

The pressure drop through a pipeline carrying a flashing fluid is calculated. A two-phase, non-flashing mixture may also be considered. The program uses the Dukler correlation to determine the pressure drop of the mix after calculating the liquid and vapor pressure drops. The pipe is divided into foot-long segments, and a separate calculation is done on each foot, thus approaching an exact solution. Adiabatic or non-adiabatic pipelines may be considered.

VISCOS (Viscosity of a Multicomponent Mix)

The method of Wilke is used to calculate the viscosity of a multicomponent mix, given the individual viscosities and amounts. The program accepts engineering input, e. g., degrees F, pounds of a component, and viscosity in centipoises, and outputs the viscosity in centipoises.

THCOND (Thermal Conductivity of a Multicomponent Mix)

The method of Lindsay and Bromley is used to calculate the thermal conductivity of a multicomponent mix. The program accepts engineering units.

DIFFUS (Binary Diffusion Constants of a Multicomponent Mix)

The method of Hirschfelder, Bird and Spotz is used to calculate the individual binary diffusion constants in a multicomponent mix.

Example:

MCCABE-THIELE CALCULATIONS

(1) MOL FCT X IN DISTLT = .98
(2) MOL FCT X IN FEED = .17
(3) MOL FCT X IN BOTMS = .005
(4) QUALITY OF FEED = .8
(5) REFLUX RATIO = 2.77

DO DATA PTS COME FROM FILE? NO

INPUT THE EQUILIBRIUM PTS

	Y	X
POINT 1 =	0.000	0.000
POINT 2 =	.6.	.042
POINT 3 =	.828.	.250
POINT 4 =	.932.	.900
POINT 5 =	.828.	.250

PT LESS THAN LAST. TRY AGAIN
POINT 5 = .932..956

PT LESS THAN LAST. TRY AGAIN
POINT 5 = 1.1

TABULATION OF INPUT POINTS

X	Y
0.000	0.000
0.042	0.600
0.250	0.828
0.900	0.932
1.000	1.000

X	Y	STAGE
---	---	-------

ENRICHING SECTION

0.971	0.980	1
0.960	0.973	2
0.949	0.966	3
0.938	0.958	4
0.925	0.949	5
0.911	0.939	6
0.883	0.929	7
0.752	0.908	8
0.236	0.813	9

STRIPPING SECTION

0.030	0.433	10
0.020	0.282	1
0.003	0.048	2

TOTAL # ENRICHING STAGES = 10

TOTAL # STRIPPING STAGES = 2

TOTAL NUMBER OF STAGES = 12

EQUATIONS OF THE TWO LINES

Y = 0.735X + 0.260 (ENRICH)
Y = 2.897X - 0.009 (STRP)
PT OF INTSECTN OF THE TWO LINES

(0.352,0.125)

WANT TO RERUN? NO

WANT TO SAVE DATA PTS? YES

FILE NAME = MCCABE

The programs described here are based in large part upon the respected Dartmouth BASIC Library created by the originators of the BASIC language. They find broad application as utility routines and subroutines; as demonstration programs; as "starter" programs for teaching, and as a point of departure in new programming. New routines are added periodically. Your suggestions, or contributions of new programs of general interest will be welcomed.

Programs are accessed as follows:

•BASIC
NEW OR OLD--OLD
OLD FILE NAME--ACCELR***

READY

Three asterisks (***) following the program name ACCELR, instructs the System to make a library search for the program and make it available. No period (.) should be used between program name and asterisks.

The operating instructions are accessed from monitor mode (before BASIC) as follows:

TYPE [0-1-1100] ACCELR.OPR

DEMONSTRATION PROGRAMS

ACCELR***	Calculates the time required for a car to accelerate from zero to 60 mph.	JFK***	Prints a picture of John F. Kennedy.
AMAZNG***	Constructs a different maze every time it is run—Guarantees only one path through.	SNOOPY***	Prints a picture of Snoopy.
CALEND***	Determines the day of the week on which a given date falls.	XMAS***	"The Twelve Days of Christmas" sing-along.

ENGINEERING PROGRAMS

BEMDES***	Recommends the correct steel beam to use for a number of common applications.	HCROSS***	Solves the flow of water through a network of pipes by the "Hardy Cross" method.
CONVRT***	Converts measurements of temperature, length, area, and density (of pattern) from one scale to another.	LPFILT***	Designs low pass filters using constant K prototype T section and M derived ($M = 0.6$) termination L sections.
DEBYE***	Computes the Debye or the Einstein function.	PAD***	Takes the arithmetic out of attenuator pad design. Six types of pads are designed.
DISTIL***	Simulates a binary batch distillation in a packed tower. Constants supplied by user.		

(continued on pg. 2)

* BASIC was developed by Dartmouth College, Hanover, New Hampshire, and is copyrighted by the Trustees of Dartmouth College. These programs are in part an adaptation of Dartmouth programs.

FINANCIAL PROGRAMS

ACCRUL***	Calculates accrual of interest on installment loans.	INSTL1***	Calculates the monthly payments schedule for an installment loan—loan statement format.
ANUIT1***	Determines how long a particular amount of money will last when invested at a particular rate of interest, with periodic withdrawals.	LESSEE***	Compares lease and purchase of equipment, using the Bower-Williamson method of analysis.
ANUIT2***	Performs calculations for payment and withdrawal annuities.	LESSIM***	Calculates the lessor's cash flows and rate of return on a lease with period-end payments.
BNKPRJ***	Projects eight balance sheet and income statement entries up to ten years into the future.	LESSOR***	Same as LESSIM but with sensitivity analysis.
BNKSIM***	Simulates one year's activities of a small bank using probabilities for withdrawals and deposits.	MAKBUY***	Calculates the present value of the cost saving of making a product as opposed to buying it.
BONDPR***	Computes the price and accrued interest for a bond.	MORTCA***	Computes Canadian mortgage table.
BONDYD***	Computes the yield to maturity of a bond, before and after tax.	MORTGE***	Computes standard mortgage table.
CAPBUD***	Capital budgeting—compares up to 4 projects, and gives internal rate of return.	NICK63***	Stock analyzing program—computes the growth rate required to justify the price of a stock.
DEMAND***	Calculates interest due for each demand loan outstanding at the end of a 3-mo. quarter.	POPULA***	Projects population growth for any number of years using the compound interest formula.
DEMSEC***	Prints a schedule of demand loans broken down by interest rate.	RCONCL***	Reconciles a customer's ending bank statement balance with those checks and deposits that are outstanding.
DEPREC***	Computes and prints depreciation by monthly summaries using four different methods.	RESERV***	Calculates Federal Reserve position.
EDKUH1***	Computes one or more multiple regressions on National Income Account data.	RETURN***	Computes a matrix of returns for an investment in a stock.
FRECAP***	Forecasts balance sheets, income statements, and cash flow statements for 5 years.	SALES***	Prints out information to assist a department store.
FRECS4***	Computes Henri Theil's measures of quality of forecast.	SAVING***	Calculates the amount of money that would accumulate after N years at an annual interest rate R compounded T times per year.
INCTAX***	Determines personal income tax according to Form 1040 using the 1968 rates.	SBA***	Computes and prints out a monthly schedule for a SBA loan.
INOUT***	Does input/output analysis for a hypothetical economy.	SWITCH***	Computes the net effect of switching from a bond currently held to another bond under consideration.
INSTAL***	Calculates the monthly payments schedule for an installment loan—detailed ledger format.	TAXRA***	Calculates the discounted rate of return, on an after-tax basis, of an investment.
		TRUINT***	Calculates the true annual interest rate charged on an installment loan.

GAME PROGRAMS

BANDIT***	Simulates a slot machine (the One-Armed Bandit).	DIGITS***	Guesses a sequence of numbers.
BASBAL***	Simulates a baseball game.	FOOTBL***	Simulates a football game.
BATNUM***	Battle of numbers game.	GOLF***	Plays golf.
BINGO***	Game of bingo.	HNGMAN***	Plays hangman.
BLKJAK***	Plays a game of blackjack—computer is dealer.	HOSSR***	Simulates a horse race.
BRIDGE***	Bridge bidding practice.	LEARN***	Learns to play the game of "21".
BSKBAL***	Game of basketball.	NIM***	Ancient game of nim.
CRAPS***	Simulates a crap game.	NIM2***	Game of nim (more options than NIM***).

(continued on pg.3)

GAME PROGRAMS *Continued*

POKER***	Plays 5-card draw poker.	ROULET***	Plays roulette.
QUBIC***	Plays tic-tac-toe in a 4x4x4 cube.	TICTAC***	Plays tic-tac-toe.
QUEEN***	Plays the queen game (game based on queen's move in chess).	WAR***	Card game of war.

MATHEMATICAL PROGRAMS

4SQRS***	Writes integers as a sum of the squares of four integers.	DERIV***	Finds the derivative of a function at a point.
ARITH***	Performs arithmetic with numbers of up to 300 digits; it adds, multiplies, or divides.	DESYS***	Solves the initial value problem: $X' = F(T, X, Y)$ $Y' = G(T, X, Y)$ $X(T_0) = X_0$ $Y(T_0) = Y_0$
ASGNMT***	Solves the classic assignment problem and computes a cost for the assignment.	DGMINV***	Finds the value of Z to make $D(Z) = D_0$, where $D(Z)$ is the digamma function of real Z , $Z > 0$.
BESSEL***	Defines a function for calculating Bessel functions of the first kind.	DIFEQ***	Solves a set of N first order differential equations using the fourth-order Runge-Kutta method.
CDETER***	Evaluates the determinant of a complex matrix, using the Crout method.	DIGAMA***	Finds digamma of real Z , $Z > 0$.
CHINES***	Chinese remainder theorem—solves simultaneous congruences of the form: $A(I) \cdot X = B(I) \text{ MOD } M(I)$, FOR $I = 1$ TO N , $N \leq 100$	DIVIDE***	Synthetic division—finds $FNN(X)/FND(X)$.
CIRCLE***	Divides a circle into a given number of equal parts: gives coordinates & angles.	ECHAIN***	Computes basic quantities for an ergodic Markov chain.
CMDDET***	Similar to CDETER***, except that a different algorithm is used.	EUCLID***	Finds the greatest common divisor of two integers.
COMPLX***	Gives the values of sin, cos, tan, sinh, cosh, and tanh of complex variables.	FACTOR***	Finds the prime factorization of a number.
CONCLD***	Determines the strongest possible conclusions involving the fewest possible variables from given sentences of propositional logic.	FALSE***	Useful in obtaining an implicit solution.
CPM***	Critical path analysis.	FIB***	Locates and evaluates the maximum of a unimodal function of one variable.
CRCPM***	Creates the necessary input for CPM***	FNZERO***	Looks for one zero of a function in a given interval.
CROUT***	Solves M sets of N linear equations in N variables. Using the Crout algorithm with row interchanges.	GREGRY***	Uses Gregory's method of integration.
CSQRT***	Calculates square roots of complex numbers.	HKJVS***	Does a Hooke-Jeeves pattern search.
DE1OR***	Solves the initial value problem: $Y' = F(X, Y)$ $Y(X_0) = Y_0$	HUGPRI***	Searches for primes in a specified range of 1000.
DE2OR***	Solves the initial value problem: $Y'' = F(X, Y, Z')$ $Y(X_0) = Y_0$ $Y'(X_0) = Y_0'$	INTGRT***	Simple integration routine.
		INVERS***	Inverts a matrix using the exchange method.
		INVHIL***	Contains a routine which produces the inverse of an $N \times N$ segment of the Hilbert matrix.

(continued on pg.4)

MATHEMATICAL PROGRAMS *Continued*

LEGEN***	Evaluates the Legendre symbol (A/P).	ROOTER***	Finds the roots of polynomials using Bairstow's method.
LINEAR***	Solves a system of linear equations.	SIEVE***	Demonstrates the sieve method of finding primes.
LINPRO***	Linear programming using the two-phase simplex method.	SIMEQN***	Solves a system of linear equations with sets of right-hand-side (constant) values.
LINQUD***	A linear or quadratic programming algorithm.	TAUTLG***	Determines if a formula of the propositional calculus is a tautology. Input - data statements.
MEALY1***	Determines the classes of equivalent states of any specified finite-state Mealy machine.	TCHAIN***	Computes basic quantities for a transient Markov chain.
MEALY2***	Simulates the I/O behavior of any specified finite-state Mealy machine with initial state.	TRIANG***	Finds the unknown features of any triangle, given one side and 2 other parts.
NTRUTH***	Computes N-valued truth tables for formulas of the propositional calculus.	TTLG***	Determines if a formula of the propositional calculus is a tautology. Input - requested.
NUMINT***	Evaluates definite integrals using Gauss' rule	TURMUL***	Simulates the action of a Turing machine.
PERT***	Analyzes a PERT network.	WELLFM***	Checks to see if a string of symbols constitutes a well-formed formula of the propositional calculus.
PHICOE***	Computes the value of the phi coefficient and the number of cases for data on 2 variables.	YIELD***	Analyzes a Markov chain with 2 absorbing states and N transient states.
PYTHAG***	Generates 86 primitive triples.	ZEROES***	Locates "interesting" values of X for any function F(X).
RATINV***	Inverts a matrix whose elements are expressed as ratios of two numbers.		
REVSIM***	Revised simplex linear programming.		
ROMINT***	Performs Romberg's integration on any function F(X).		

MISCELLANEOUS PROGRAMS

FORME***	Prints a number in the Fortran E format (easily modified to be used as a subroutine).	PLOT***	Plots any number of points (X, Y).
FORMF***	Prints a number in the Fortran F format (easily modified to be used as a subroutine).	PLOTR***	Plots a maximum of 100 points (may be entered in any order).
FORMI***	Prints a number in the Fortran I format (easily modified to be used as a subroutine).	PLOTTO***	Plots one to six functions of X on the same graph.
GRAPH***	Plots the graph of a function.	SORT***	Sorts entries with 4 terms, 2 alphabetic and 2 numeric.
GRAPH1***	Plots a function.	TWOPLO***	Simultaneously plots two functions of a single variable X.
HSTGRM***	Plots a histogram of test scores.	XYPLOT***	Plots single-valued functions of X (with X on the vertical axis).
INDEX***	Makes an index by grouping together identical entries (& page numbers), and alphabetizing.		

STATISTICAL PROGRAMS

2222***	Calculates and displays the decomposition of Q for a four variable table.	BACT2L***	Bayesian analysis of a 2-level contingency table.
2X2***	Calculates percentage tables, chi-square value, probability, and Q for 2x2 tables.	BACT3L***	Bayesian analysis of a 3-level contingency table.
2X2X2***	Calculates percentages, Yule's Q, and confidence intervals for a dichotomized 3-variable table.	BAYES***	Revises probabilities according to Bayes' theorem.
ANOVAR***	Analysis of variance for a factorial design of up to 14 factors.	BICONF***	Computes population proportion and confidence limits, based on the binomial distribution.
		BINODI***	Calculates binomial distribution.

(continued on pg.5)

STATISTICAL PROGRAMS *Continued*

BINOMC***	Computes binomial coefficients—N things taken 1 at a time.	EMPIRI***	Draws random samples from a distribution defined by the Pth, 50th, and (100-P)th percentiles.
BINOMD***	Binomial distribution—computes the probability of I successes, and I or more successes, for all I, $I \leq N$ where N is the number of trials.	FPOFLT***	Does a least squares fit to a polynomial curve.
BINOMI***	Computes values of the binomial distribution.	FVALUE***	Computes the exact probability of an F-ratio with (M,N) degrees of freedom.
BINONI***	Prints a table of binomial probabilities given the probability of success on each trial and the number of trials.	GAME***	Solves a 2x2 game theory matrix.
BINOPO***	Compares binomial probabilities with the approximations of the normal and Poisson distributions.	GAMMAT***	Solves a game theory matrix up to 15x15.
BINORM***	Uses the normal approximation to the binomial distribution. Computes $P(A \leq X \leq B)$, where X is the number of successes in N trials.	GEOMEN***	Computes the geometric mean and std. deviation for a geometrically normal set of data.
BITEST***	Does calculations necessary for a statistical test on a binomial population.	GRADER***	Calculates the term average for each student and finds the class average.
CGRADE***	Computes standard score, weighted cumulative scores, and class ranks for a class of students.	GROUP***	Uses grouped data to calculate various statistical measures.
CHISQ***	Computes chi-square values and probabilities for MxN tables. Input - data statements.	HYPERG***	Computes and prints a table of hypergeometric probabilities for a specified sample population.
CHISQX***	Computes chi-square values and probabilities for MxN tables. Input - requested.	LINCON***	Simple linear regression.
CONBIN***	Estimates population proportion and computes confidence limits, based on the normal distribution.	LINFIT***	Computes the best linear fit for a set of independent variables to a dependent variable.
CONDIF***	Computes confidence limits for the difference between two population means - normal distribution.	MANDSD***	Calculates the mean, variance, and standard deviation for individual values or distributions.
CONLIM***	Computes confidence limits for an unknown population mean, based on the normal distribution.	MONCAR***	Evaluates a function of a maximum of 16 random variables, using Monte Carlo simulation.
CORREL***	Computes a correlation coefficient.	MSTDEV***	Computes the mean, variance, and standard deviation of a set of grades.
CURFIT***	Fits 6 common curves thru points using a least squares fit.	MULFIT***	Multivariate curve fit.
DECID2***	Decides between an arbitrary judgement or a test on a piece of equipment suspected to be damaged.	NORDEV***	Generates random numbers which are normally distributed with mean 0 and variance 1.
DECIDE***	Helps decide among 3 alternatives.	ONEWAY***	Performs a one-way analysis of variance with equal sample sizes.
		PDIS***	Calculates the proportional distribution of a row (or column) of figures, and the confidence intervals for each frequency.
		POLFIT***	Fits least-squares polynomials to bivariate data, using an orthogonal polynomial method.

(continued on pg. 6)

STATISTICAL PROGRAMS *Continued*

PRBSTA***	Probability analysis - computes the probabilities of 10 statements and their denials.	STAT13***	Computes the analysis of variance table for a one-way completely randomized design.
PROVAR***	Does calculation for normal or student's T distribution.	STAT14***	Produces the analysis of variance table, F-ratios for treatments and blocks of a randomized complete block design.
RANDIG***	Produces 500 random integers between 0 and 99.	STAT15***	Computes the analysis of variance table for a simple Latin square design.
RANDOM***	Randomly orders the integers from 1 to N.	STAT16***	Computes the analysis of variance table for a simple Graeco-Latin square design.
ROW***	Calculates the proportional distribution of a row or column of figures and 95% confidence intervals.	STAT17***	Computes the analysis of variance table and the F-ratio for treatments for a balanced incomplete block design.
RXC***	Computes chi-square, percentage table, expected cell values, and gamma for a table of data.	STAT18***	Computes the analysis of variance table and the F-ratio for treatments for a Youden square design.
SEVPRO***	Applies a chi-square test to several sample proportions.	STAT19***	Computes the slope and other statistics for a linear regression with several Y values for each X.
STAT00***	Computes the mean, variance, standard deviation and standard error of the mean.	STAT20***	Performs a multiple linear regression according to Efromson's algorithm.
STAT01***	Computes the means, standard error of means, mean differences, standard error of difference, and T-ratio for two groups of paired data.	STAT21***	Computes one or more multiple linear regressions on a batch of data.
STAT02***	Computes the means, variances, and T-ratio for two groups of unpaired data. (unequal variances)	STAT33***	Same as STAT13*** but with weighted observations.
STAT03***	Computes the means, variances, and T-ratio for 2 groups of unpaired data. (equal variances)	STATAN***	Computes 34 different measures for an array of weighted or unweighted values of one variable.
STAT04***	Computes chi-square values and probabilities for any number of two by two tables.	STATNW***	Same as STAT10*** but for weighted numbers.
STAT05***	Calculates the sign test confidence interval using fractional counts.	STATWE***	Computes statistics for a simple linear regression with one independent variable.
STAT06***	Calculates the confidence limits for a set of data using the Wilcoxon signed rank sum procedure.	STOVAR***	Contains 9 subroutines which generate random numbers according to 9 probability distributions.
STAT08***	Compares two groups of data using the median test. Produces a 2x2 table and its chi-square value.	TESTMN***	Computes the probability of finding a certain population mean, based on the student's T distribution.
STAT09***	Compares two groups of data by means of the Mann-Whitney two sample rank test.	TVALUE***	Computes the two-tail probability of a T-value.
STAT10***	Computes 34 statistical measures for a series of weighted numbers as given in the NBS handbook 101.	UNISTA***	Computes various statistics on a list of numbers.
STAT11***	Computes the Spearman rank correlation coefficient for two series of data.	WALDS***	Computes the important characteristics of Wald's sequential test procedure.
STAT12***	Computes the correlation matrix for N series of data.	WORDS***	Writes first draft of a report on a 2x2x2 table.
		XPOLFT***	Least squares fit to a polynomial curve.

AID for AL/COM

(Algebraic Interpretive Dialogue) FEBRUARY, 1970

AID (Algebraic Interpretive Dialogue) is a direct and easy-to-learn mathematical language capable of solving complex numerical problems. Its simple commands, based upon imperative English sentences, plus its interactive response to typed input, make it a particularly attractive "computing aide" for scientists and engineers, as well as those inexperienced in more formal programming languages.

Commands are typed as short imperative verbs followed by their arguments, and numerical values written, for the most part, in standard mathematical notation. Facilities for editing, the ability to save procedures, the use of function libraries on the System, and the writing of Boolean propositions are all among the features included in AID. Simple, self-explanatory diagnostics are also provided. When used with the Model 37 Teletype or IBM 2741, AID recognizes upper and lower case characters.

AID runs in 11 K of core, with 1 K of user data area, expanding to 14 K with 4 K of user data area when required.

AID COMMANDS	DESCRIPTION
CANCEL	Cancels a stopped process.
(CANCEL)	Cancels a stopped process initiated by a parenthetical DO.
DELETE ...	Erases the specified item or items from storage.
DEMAND ...	Causes type out requesting a value for a variable (1 per command).
DISCARD ITEM m	Deletes item m from external file in use.
DO ...	Causes specified iteration of a previously defined indirect step or steps. Indirect steps return control to next step.
(DO ...)	Starts a new execution without cancelling a stopped process.
DONE	Skips execution of remaining steps of a part during current iteration.
FILE ... AS ITEM ...	Stores item in open external file.
FORM	Defines a format for typeout.
GO	Continues execution of a currently stopped process.
... IF ...	Conditional command. Causes command to which it is appended to be executed only if the proposition following the IF is true.
LET ... = ...	Defines arithmetic formulas, Boolean functions, and user functions and associates them with identifiers. Formula or function is reevaluated each time identifier appears.
LINE	Advances paper one line.
PAGE	Advances paper to top of next page.
QUIT	Skips execution of remaining steps of a part and cancels further DO iterations.
RECALL ITEM ...	Reads in item stored by FILE command.
RESET TIMER	Resets control processor timer.
SET ... = ...	Assigns a value to a variable.
STOP	Halts execution and returns control to user.
TO m	Stops execution in progress and transfers control to new step m.
TYPE ...	Types out specified information.
USE FILE ...	Accesses an external storage file.

Continued on other side

*AID is an adaptation by Applied Logic Corporation of the JOSS[†] language, developed by the RAND Corporation under contract with the United States Air Force.

[†]JOSS is a trademark and service mark of The RAND Corporation for its computer program and services using that program.

AL/COM is a direct access Computer Time-Sharing Service of Applied Logic Corporation • 1 Palmer Square • Princeton, N.J. 08540 • 609-924-7800

AID Characters and numerical operations

A — Z	upper case	+	addition	/	division
a — z	lower case	-	subtraction	*	multiplication
0,1,...,9	digits	=	equal	\$	current line number
!...!	absolute value	#	not equal	*	cancel line
[]	brackets	<=	equal or less than	<	less than
()	parentheses	>=	equal or greater than	>	greater than
↑	exponentiation	←	character position in typeout formats		

EXAMPLE:*

A conditional expression can be used to perform a table lookup for all items whose values satisfy one or more conditions.

*1.1 Set $A(x) = x^2 + x \cdot 5 - 5 \cdot x$.

Formula to generate xth table item.

*Do step 1.1 for $x = 1(1)35$.

Generate a 35-item table.

*Type $A(20)$.

$A(20) = 310$

*Type $A(3)$.

$A(3) = -4.5$

*Let $F(x) = (x < 0 : x ; x > 700 : x ; fp(x) > 0 \text{ and } x > 300 : x ; +)$.

Find all values which are (1) less than zero; (2) greater than 700; or (3) greater than 300 and have a fractional part which is nonzero. If x is none of these, perform a line feed, carriage return (indicated by the + symbol).

*1.1 Type $F(A(i))$.

Test every fifth item in the table.

*Do step 1.1 for $i = 1(5)35$.

$F(A(i)) = -3.5$

Values in tested items which do not satisfy any of the three propositions result in line feed/carriage return (because of the terminating + symbol in the conditional expression).

$F(A(i)) = 346.5$

$F(A(i)) = 821.5$

$F(A(i)) = 1067.5$

*Let $E(x) = (fp(x/2) = 0 : x ; +)$.

Find all even-numbered values in the table.

*1.1 Type $E(A(i))$.

Test every item in the table.

*Do step 1.1 for $i = 1(1)35$.

$E(A(i)) = -2$

$E(A(i)) = 28$

$E(A(i)) = 90$

$E(A(i)) = 184$

$E(A(i)) = 310$

$E(A(i)) = 468$

$E(A(i)) = 658$

$E(A(i)) = 880$

BASIC for AL/COM

JULY, 1969

Applied Logic Corporation has expanded BASIC on the AL/COM System to meet customer demand. Features of **BASIC** include:

Constant Values—

&PI The value of pi
&E The value of the transcendental number "e"
&Q\$ The single character "

Catalog—

By typing **CAT** in BASIC, the user gets a listing of all BASIC files in his directory.

File Capability—

Users may read and write files in BASIC of over 1 million characters. The commands are BASIC-easy (see example on reverse side). For an explanation of BASIC's file capability on the AL/COM System, type

TYPE SYS:BASIC.MOD

String Variables are defined by a \$; thus string labels may be A\$, B3\$. A list of character strings is written in vector form, e.g. V\$ () signifies a list of strings and V\$ (3) is the third string in the list.

All **Variables** are automatically set to Ø before each run.

In **Looping**, the initial, final, and incremental indices may be formulas of any complexity. Also, the initial index may be greater than the final with a decremental index, e.g.

7Ø FOR I = (C1/.4) TO -6.1 STEP -.81Ø

TAB function on output for formatting, e.g.,

195 PRINT "CREDIT" TAB(2Ø) "DEBIT" TAB(45) "TOTAL"

200 PRINT C1 TAB(2Ø) D1 TAB(45) T2

13 **Matrix** instructions for transposing, inverting, multiplying, finding the determinant, and 8 other standard matrix operations.

Library programs may be referenced as old files by adding three asterisks to the name, e.g.:

NEW OR OLD—OLD

OLD FILE NAME—MORTCA * * * calls the mortgage calculation program and loads it.

Additional modification to BASIC will also be recorded in SYS: BASIC.MOD as they are implemented.

More on other side

Create data file for input. BASIC

Program expects a file name extension ".BAF" in addition to the primary file name.

```
.CREATE INPT.BAF
```

```
*I10
```

```
00010 1.5,-4.278
```

```
00020 "THIS IS A SAMPLE OF READING & WRITING FILES IN BASIC"
```

```
00030 $
```

```
*E
```

```
EXIT
```

```
↑C
```

Create main program in BASIC

```
.BASIC
```

```
NEW OR OLD--NEW
```

```
NEW FILE NAME--TEST
```

```
READY
```

```
10 FILES INPT,OUTFIL
```

```
11 SCRATCH #2
```

```
20 READ #1,A,B,C$
```

```
30 C=(SIN(A)/.5)
```

```
35 D=(4*B)+2/(2*A)
```

```
36 E=ABS(A*B-1.9)
```

```
40 WRITE #2,C$
```

```
41 WRITE #2
```

```
42 WRITE #2,"THE VALUES FOR C,D, AND E ARE ";
```

```
47 WRITE #2,C;D;E
```

```
60 PRINT "ALL DONE!"
```

```
99 END
```

```
SAVE
```

```
READY
```

```
RUN
```

```
BASIC
```

```
16:08
```

```
03/13/69
```

```
ALL DONE
```

```
READY
```

OUTFIL.BAF contains data generated by main program, BASIC. Use Monitor command TYPE:

```
SYSTEM
```

```
EXIT
```

```
↑C
```

```
.TYPE OUTFIL.BAF
```

```
OUTFIL.BAF 3/13/69 1616 400-1-124
```

```
THIS IS A SAMPLE OF READING & WRITING FILES IN BASIC
```

```
THE VALUES FOR C,D, AND E ARE 1.99499 97.6068 8.317
```

```
EXIT
```

```
↑C
```

The FILES statement identifies the first named data file as #1, the second named as #2, etc., up to 9 files; e.g. INPT.BAF is #1; OUTFIL.BAF, or #2, is created during program RUN. *The output file must first be SCRATCHed (line 11) before writing on it.*

Read from INPT.BAF (#1) 2 variables (A,B) and a string (C\$).

Write C\$ onto output file (#2).

Write a blank line onto output file (#2)

Write onto OUTFIL (#2) a label followed by answers (C,D,E).

Program tells us we are done; we will now leave BASIC and TYPE answers.

System command exits from BASIC.

COGO-10^{*}

AL/COM

NOVEMBER, 1968

COGO-10 is a civil engineering problem-oriented language that can be used in the solution of such geometric problems as:

1. control, land, and right-of-way surveys;
2. highway and interchange design;
3. construction layout;
4. bridge geometry.

COGO-10 has these new features:

1. programs can be written by the engineer in a matter of minutes;
2. no computer programming knowledge required;
3. programs can be of any length—not limited by size of computer memory;
4. COGO-10 programs are written using a series of commands which describe basic civil engineering operations such as ALIGNMENT (optional short form ALN) used to calculate the horizontal alignment of a curve;
5. data required for each operation is included as part of the command, and free format is permitted;
6. known and calculated coordinates may be saved and used up to 999 points;
7. calculated distances and angles may be saved and used;
8. comments may be inserted anywhere;
9. COGO-10 has format control of output which is easy to read and associate with the corresponding input data.

These features combine to make COGO-10 a system that is both easy to use and extremely efficient.

COGO-10 and AL/COM combine to decrease the communication barrier between the civil engineer and the computer.

COGO-10 and AL/COM free the civil engineer from tedious computation chores and permit more of his time to be devoted to true engineering work.

Sample Problem and a COGO-10 Solution on other side:

**COGO-10 is the time-sharing version of COGO-90, which was developed jointly by the Puerto-Rico Department of Public Works and the Department of Civil Engineering, M.I.T. COGO-10 is 100% compatible with COGO-90.*

Sample Problem and a COGO-10 Solution

Known Data:

- coordinates of points 6 and 51
- length and azimuth of line 51-22
- length and bearing of line 22-14
- length of line 51-97 and angle at 51 from 22 to 97.

Desired Results:

- coordinates of all points
- length and bearing of line 14-97
- angle at 97 from 14-51
- area of the parcel bounded by side 35-42, arc 42-342, side 342-22, and side 22-35 and the length and azimuth of the sides of the parcel.

COGO-10 Solution: (What you type on the remote console is underlined)

After logging in

↑C

.R COGO

COGO-10 is ready. Type "Disk/File Name." If commands are on disk otherwise type in commands on remote console.

:CLEAR 1 999

:STORE 51 5000.00 2000.00

:LOCATE AZIMUTH 51 22 2236.07 116 34 00.

PT 22, N=3999.943, E=3999.974

:LOCATE/BEARING 22 14 2828.67 1 45 15 39.

PT 14, N=5990.988, E=6009.230

:*POINT 14 ON MAIN STREET

:LOCATE/ANGLE 22 51 97 3605.09 -82 27 00.

PT 97, N=7984.644, E=4022.022

:INVERSE/BEARING 97 14

PT 97 TO PT 14, DIST=2814.900, BRG=S 44 54 25.902 E

:ANGLE 14 97 51

ANGLE BY PTS: 14- 97- 51=79 1 25.886

:STORE 6 8000.00 6000.00

:POINTS/INTERSECT 42 51 6 14 97

PT 42, N=7144.550, E=4859.399

:LOCATE/LINE 51 42 35 1156.21

PT 35, N=5693.726, E=2924.968

:DISTANCE/SAVE 51 42 20

PT 51 TO PT 42, DIST=3574.249

:ARC/LINE/POINTS 342 51 D20 14 22 14

PT 342, N=5521.977, E=5535.930

:AREA/AZIMUTHS 35 42 342 22 35

PT 35 TO PT 42, DIST=2418.039, AZ=53 7 48.369

PT 42 TO PT 342, DIST=1757.963, AZ=157 21 58.854

PT 342 TO PT 22, DIST=2162.348, AZ=225 15 39.008

PT 22 TO 35, DIST=2006.125, AZ=327 35 51.521

AREA=4179026.700 SQ. FT. and 95.937 ACRES

:*LAND PARCEL OWNED BY J. SMITH

:SEGMENT/PLUS 42 342 D20

CHORD=1757.963, ARC=1776.183, SEG AREA=129042.520 SF

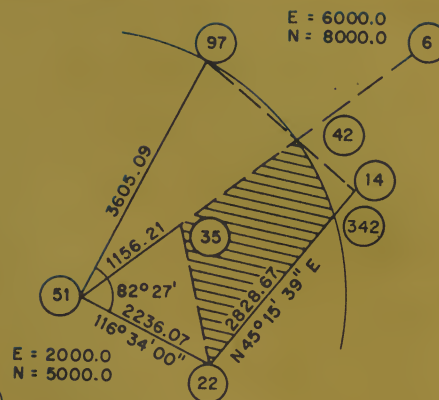
CUM AREA=4308069.300 SF OR 98.900 ACRES

:READ/CONSOLE

:FINISH

EXIT

↑C



ECAP for AL/COM

(Electronic Circuit Analysis Program) DECEMBER, 1969

ECAP (Electronic Circuit Analysis Program) is an integrated system of programs developed to aid the electrical engineer in the design and analysis of electronic circuits. There are four closely related programs which produce DC, AC or transient analyses of electrical networks from a source program description of circuit topology, a list of corresponding component values, a selection of the type of analysis desired, a description of circuit excitation, and a list of output desired.

ECAP is not difficult to use. It is not necessary for the user to have any knowledge of the internal mechanics of the program or any prior programming experience. The techniques involved in using ECAP are easily learned by the electrical engineer because ECAP's input language and its output are written in electrical circuit terminology.

Advantages of AL/COM ECAP: One might consider ECAP as an electronic circuit simulator which permits the designer to use capabilities of the digital computer. ECAP has several advantages over breadboarding:

First, the effects of parameter variations can be much more easily observed using a computer program than using breadboard evaluation.

For example: the effects of changing only collector capacitance or alpha cutoff frequency of a transistor are difficult to determine experimentally, but are easy to evaluate with the computer.

Second, the circuit designer can analyze the effect of a component failure on circuit performance without removing or destroying the component involved.

Third, the computer allows the designer to study combinations of component parameters, such as worst case, which are difficult if not impossible to achieve in the laboratory, and high power circuits which would otherwise require construction of expensive small scale models.

Fourth, the program makes it possible to simulate the effects of expensive or hard-to-obtain components.

ECAP capabilities are outlined in the following description of the four modules available to the user.

Input Language: This program acts as the communication link between the user and the three analysis programs. The language is user-oriented and allows complex circuits to be simply described to the computer from easily developed equivalent circuits. ECAP language statements are used to completely define the topology of the circuit, circuit element values, type of analysis to be performed, driving functions, and output required.

DC Analysis: The DC analysis program obtains the DC or steady-state solutions of linear electrical networks and provides the worst case analysis, standard-deviation (statistical) analysis, and sensitivity coefficient calculations if requested. This program also provides an automatic parameter modification capability.

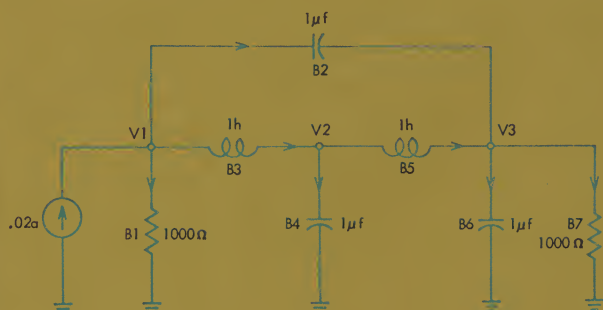
AC Analysis: The AC analysis program obtains the steady-state solution of linear electrical networks subject to sine-wave excitation at a fixed frequency. Since this program also contains the automatic parameter modification capability, it is easy to obtain frequency response and phase response solutions.

Transient Analysis: The transient analysis program provides the time-response solution of linear or nonlinear electrical networks subject to arbitrary, user-specified driving functions. Nonlinear elements are modeled by using combinations of linear elements and switches to provide piecewise linear approximations to the nonlinear characteristics.

Continued on other side

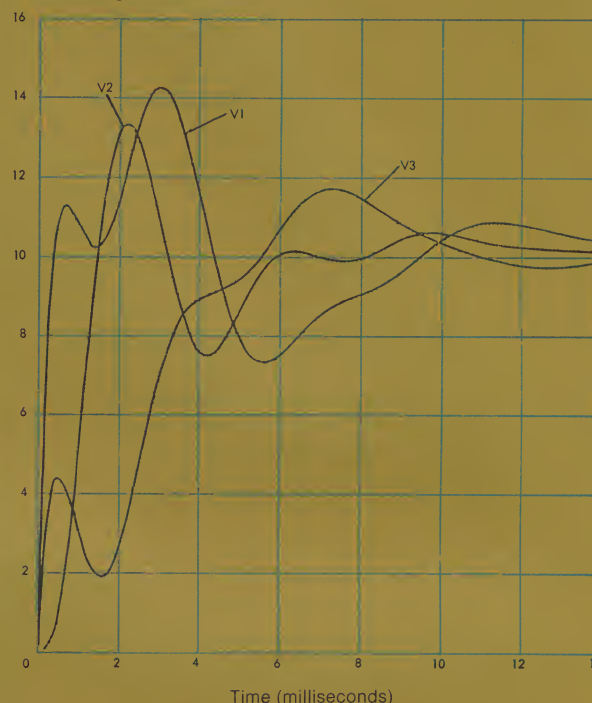
An Example of ECAP Analysis

A sample ECAP analysis of a "bridged T" circuit is reproduced here to illustrate the extreme simplicity and speed of this user-oriented program. Transient analysis of circuits of this general type by conventional means is a time-consuming procedure and usually involves undesirable approximations. After creating a data file with ECAP statements (in this case TR3) the user operates ECAP and responds to three basic program control questions.



Circuit Schematic

Node Voltages at V1, V2 and V3 (volts)



Bridged T circuit transient response
(plotted from ECAP results)

AL/COM ECAP 10/ 6/69 11:10 AM

INPUT = TR3

OUTPUT = TTY

LIST INPUT FILE? YES

```

00100      TRANSIENT ANALYSIS
00110      B1      N(1,0),R=1000,I=.02
00120      B2      N(1,3),C=1 E-6
00130      B3      N(1,2),L=1
00140      B4      N(2,0),C=1 E-6
00150      B5      N(2,3),L=1
00160      B6      N(3,0),C=1 E-6
00170      B7      N(3,0),R=1000
00180      TIME STEP =.01 E-3
00190      OUTPUT INTERVAL = 50
00200      FINISH TIME = 10E-3
00210      PRINT, VOLTAGES, CURRENTS
00220      EXECUTE
    
```

TRANSIENT ANALYSIS RESULTS:

T = 0.0000000

NODES NODE VOLTAGES

1- 3 0.3999900E-03 0.5999840E-12 0.1999940E-03

BRANCHES ELEMENT CURRENTS

1- 4 0.3999900E-06 0.1999960E-01 0.3999900E-10 0.5999840E-10
 5- 7 -0.1999940E-10 0.1999940E-01 0.1999940E-06

T = 0.5000000E-03

NODES NODE VOLTAGES

1- 3 0.1038332 E+02 0.8831565 E+00 0.4308734 E+01

BRANCHES ELEMENT CURRENTS

1- 4 0.1038332 E-01 0.6471922 E-02 0.3144762 E-02 0.4493564 E-02
 5- 7 -0.1348802 E-02 0.8143865 E-03 0.4308734 E-02

GASP^{*} for **AL/COM**

(General Activity Simulation Program) FEBRUARY, 1970

AL/COM GASP II is a versatile and easy-to-learn Fortran IV based simulation language useful in simulations of discrete events and construction of queueing, network modeling and similar programs. GASP finds wide application in the fields of operations research, systems analysis, production and inventory control, and many other areas of study.

CONVERSATIONAL MODE

GASP, as implemented on AL/COM, has been adapted to take advantage of its many convenient and time-saving features, and to operate in a conversational mode, facilitating program preparation and debugging, as well as changes in program data or functions between simulation runs.

A conversational editor is used to process standard GASP variables and verify input data. The user may examine pertinent GASP program controls at any time and a debug mode is provided which types out diagnostic messages in the event of error.

GASP LIBRARY STRUCTURE

All standard GASP subprograms and utility programs are included in a single AL/COM library. They are incorporated into the user's FORTRAN written simulation program by the appropriate FORTRAN call statements contained in a control program prepared by the user.

SPECIAL SUBROUTINES

In addition to the standard GASP subroutines described in the literature, and familiar to users, AL/COM GASP includes several subprograms which facilitate input and output file manipulation, data collection and program controls.

GASP LITERATURE

The user is referred to the following sources for detailed information on the GASP language and simulation examples utilizing GASP.

1. Simulation with GASP II, A.Pritsker and P.Kiviat, Prentice-Hall, Inc., 1969.
2. System Simulation, Geoffrey Gordon, Prentice-Hall, Inc., 1969.
3. GASP -- A General Activity Simulation Program, Applied Research Laboratory, Monroeville, Pa., United States Steel Corp., 1963
4. GASP II Program Bulletin No. 95880 (short bulletin describing implementation on AL/COM Systems), Contact your AL/COM representative, or Technical Publications Dept., Applied Logic Corp., One Palmer Square, Princeton, N.J. 08540

OTHER FEATURES

- Since GASP is based upon FORTRAN, AL/COM's extensive library of FORTRAN subroutines is particularly valuable in creating simulation functions and simulation subprograms.
- The AL/COM SEDIT, COPY, and LOAD commands, and DDT mode streamline preparation of main and control programs for GASP users.
- AL/COM's GASP II software includes subprograms to set up input/output files, collect user data, manipulate storage files and generate event data.
- For the user's convenience, and to avoid errors, a special file, COMMON.GSP, is available, containing all the basic requirements for BLANK COMMON required in GASP programs.

Continued on other side

*GASP II is a development by Arizona State University of the original GASP language created at U.S. Steel Corporation.

AL/COM is a direct access Computer Time-Sharing Service of Applied Logic Corporation • 1 Palmer Square • Princeton, N.J. 08540 • 609-924-7800

EXAMPLE:

A portion of the input, data, intermediate results and summary report for a typical GASP simulation problem, run on an AL/COM System, is illustrated here. The user is referred to reference 1, pages 142-152, listed under "GASP Literature" on other side, for a complete discussion of the program and problem (a simulation of a drive-in bank).

AL/COM GASP 6-FEB-70 13:02

INPUT = TTY:

OUTPUT = TTY:

ENTER USER DATA* .4 1 1 1 1 6

START GASP DATA

*NAME JOHN DOE

*NPROJ 4

*NRUNS 1

*NCLCT 2

*NSTAT 3

*ID 20

*IM 4

*KRANK 1 4 4

*IN N 1 1 1

*JCLR 1

*NORPT 1

*TBEG 0

*TFIN=200

*JSEED=-1

*ATTRIB= .1 1. .1
WHICH FILE? *1
SCALE NOT DEFINED
ENTER SCALE VALUE*100

*FILE 1/ 1 2

*FILE 1/ 1 3

*FILE 1/ 300 4

*FILE 2 0 0

*FILE 2

*FILE 3

*FILE 3

*NDAY 6

*MON 2

*NYR 1970

*DUMP

GASP DATA DUMP

ID VALUE= 20
IM VALUE= 4
MSTOP VALUE= 0
MXC VALUE= 0
NCLCT VALUE= 2
NHIST VALUE= 0
NOQ VALUE= 3
NORPT VALUE= 1
NPRMS VALUE= 0
NRUN VALUE= 1
NRUNS VALUE= 1
NSTAT VALUE= 3
SCALE VALUE= 100.000
JSEED VALUE= -1
TBEG VALUE= 0.000
TFIN VALUE= 200.000
INIT VALUE= 0
ATTRIB VALUE= 0.000 0.000 0.000 0.000
INN VALUE= 1 1 1 0
KRANK VALUE= 1 4 4 0
NCELS VALUE= 0 0 0 0
PARAM PRINTOUT NOT ALLOWED
NAME PRINTOUT NOT ALLOWED
NPROJ VALUE= 4
MON VALUE= 2
NDAY VALUE= 6
NYR VALUE= 1970
JCLR VALUE= 1
*

*TYPE ERX

NAME NOT ALLOWED

*TYPE ID

VALUE= 20

*NSTAT = 345

ILLEGAL NUMERIC CHARACTER

*EXECUTE

SIMULATION PROJECT NO. 4 BY JOHNDOE

DATE 2/ 6/1970

RUN NUMBER 1

SCALE = 100.0000
CONTINUE?-ANS. "Y" OR "N"*Y

GASP JOB STORAGE AREA DUMP AT 0.0000 TIME UNITS

1	10	100	10	0	2	9999
2	100	200	0	0	3	1
3	100	300	0	0	4	2
4	30000	400	0	0	7777	3
5	0	0	0	0	6	9999
6	0	0	0	0	7777	5
7	0	0	0	0	8	9999
8	0	0	0	0	7777	7
9	0	0	0	0	10	9999
10	0	0	0	0	11	9
11	0	0	0	0	12	10
12	0	0	0	0	13	11
13	0	0	0	0	14	12
14	0	0	0	0	15	13
15	0	0	0	0	16	14
16	0	0	0	0	17	15
17	0	0	0	0	18	16
18	0	0	0	0	19	17
19	0	0	0	0	20	18
20	0	0	0	0	8888	19

GASP SUMMARY REPORT

SIMULATION PROJECT NO. 4 BY

DATE 2/ 6/ 1970 RUN NUMBER 1

GENERATED DATA

CODE	MEAN	STD. DEV.	MIN.	MAX.	OBS.
1	2.6851	2.0120	0.0000	10.9400	568
2	0.5280	0.5260	0.0000	3.9200	568

TIME GENERATED DATA

CODE	MEAN	STD. DEV.	MIN.	MAX.	TOTAL TIME
1	0.9469	0.2242	0.0000	1.0000	300.0000
2	0.9151	0.2787	0.0000	1.0000	300.0000
3	5.1036	2.4033	0.0000	8.0000	300.0000

FILE PRINTOUT, FILE NO. 1

AVERAGE NUMBER IN FILE WAS 2.8620
STD. DEV. 0.4131
MAXIMUM 4

FILE CONTENTS

300.1400	2.0000	298.9600	298.9600
300.1600	1.0000	300.1600	0.0000
300.4300	3.0000	296.1500	298.7100

FILE PRINTOUT, FILE NO. 2

AVERAGE NUMBER IN FILE WAS 1.7090
STD. DEV. 1.1454
MAXIMUM 3

FILE CONTENTS

299.5000	1.0000	299.3900	299.3900
----------	--------	----------	----------

FILE PRINTOUT, FILE NO. 3

AVERAGE NUMBER IN FILE WAS 1.5325
STD. DEV. 1.1029
MAXIMUM 3

FILE CONTENTS

300.1600	1.0000	299.5000	299.5000
----------	--------	----------	----------

MEAN TIME BETWEEN ARRIVALS = 0.40
MEAN SERVICE TIME FOR TELLERS = 1.00

PERCENT OF CUSTOMERS BALKING = 21.28
NUMBER OF CUSTOMERS BALKING = 153.00
TOTAL CUSTOMERS = 719.00

EXIT
1C

LISP 1.6

for AL/COM

(LIST Processing Language) NOVEMBER, 1969

LISP* is a versatile, interactive computer programming language, which possesses great power and flexibility for the completely recursive manipulation of symbolic data in the form of tree-structured lists. It has important applications in symbolic calculations in differential and integral calculus, high-energy physics, mathematical logic and formal proofs, game theory, language translation and artificial intelligence.

Advantages of LISP 1.6

In addition to the usual LISP features, several new features have been included, such as a compiler; an arbitrary precision integer package; an S-expression editor; up to 14 active input-output channels; the ability to control the size of memory spaces; a standard relocating loader to load assembly language or compiled programs.

Interpreter and Compiler

This LISP system uses an interpreter; however, there is also a compiler which produces machine code. Compiled functions are approximately ten times faster and also take less memory space.

Arbitrary Precision Integers

LISP numbers have always been second-class citizens, in the sense that unlike strings (print names) numbers have had a maximum length. In the LISP 1.6 system there is an optional arbitrary precision integer package which extends the length of LISP integers from 36 bits to any length.

Editor

A new LISP editor is now available. Significant improvements have been made in the command structure and editing speed. The major addition is a pointer, which can be moved through the editor's string, the editing features affecting only the area to the right of this pointer. One can insert and delete arbitrary character strings, and file and unfile these strings on various I-O devices.

Memory Allocation

The LISP 1.6 system has memory allocation and reallocation procedures which allow users to adjust the sizes of the various areas of memory to their own needs.

Loader

A modified version of the standard PDP-6/10 MACRO-FORTRAN loader is available for use with LISP. One can call the loader into LISP core image at any time to combine LISP-compatible MACRO programs.

AL/COM Documentation

A complete LISP 1.6 Reference Manual, Number RM9078, is available from the Technical Publications Department, Applied Logic Corporation, One Palmer Square, Princeton, N. J. 08540

Continued on other side

* The LISP language and its first implementation were developed by a group under John McCarthy at Project MAC, the Massachusetts Institute of Technology, under sponsorship of the Advanced Research Projects Agency. The PDP-10 version was developed at the Artificial Intelligence Laboratory, Stanford University Computer Science Division, by John R. Allen and Lynn H. Quam.

Example

A small portion of a LISP demonstration program, ELIZA, designed to simulate a psychiatrist's responses to patient's (user's) input, is reproduced below, together with a typical session on the couch.

```

TC
JR ELIZA
(WORKER)
HELLO
HOW DO YOU DO. PLEASE STATE YOUR PROBLEM
MY TROUBLE IS MY MOTHER
YOUR TROUBLE IS YOUR MOTHER

YOU ARE BEING REPETITIOUS
WHAT MAKES YOU THINK I AM BEING REPETITIOUS
PERHAPS YOUR MOTHER PREFERED YOUR BROTHER
YOU DON'T SEEM QUITE CERTAIN
MY MOTHER ALWAYS LIKED MY BROTHER BEST
TELL ME MORE ABOUT YOUR FAMILY
THERE WERE TEN OF US - ALL CRAZY.
PLEASE GO ON
FATHER LEFT FOR PARTS UNKNOWN WHEN I WAS ONLY TEN.
WERE YOU REALLY
ONLY TEN?
WHAT DOES THAT SUGGEST TO YOU
THAT HE WAS LEADING A DOUBLE LIFE.
LETS DISCUSS FURTHER WHY YOUR TROUBLE IS YOUR MOTHER
SHE DOMINATES MY FAMILY.
WHY DO YOU SAY YOUR FAMILY
IT IS MY FAMILY,IS'NT IT!
DOES THAT SUGGEST ANYTHING ELSE WHICH BELONGS TO YOU
YES,MANY THINGS. THE OLD WITCH SMASHED MY CAR,FOR ONE.
EARLIER YOU SAID YOUR MOTHER ALWAYS LIKED YOUR BROTHER BEST
YES SHE PREFERRED HIM TO ME
YOU SEEM QUITE POSITIVE
YES, HE GOT THE BEST OF EVERYTHING IN THE FAMILY
BUT YOUR FAMILY
THEY SPEND TOO MUCH MONEY.
DO YOU FEEL STRONGLY ABOUT DISCUSSING SUCH THINGS
I FEEL STRONGLY ABOUT MONEY.
TELL ME MORE ABOUT SUCH FEELINGS
MY FEELING FOR MONEY DOMINATES ALL MY FANTASIES
IS IT IMPORTANT TO YOU THAT YOUR FEELING FOR MONEY DOMINATES
ALL YOUR FANTASIES
VERY! MONEY IS THE ROOT OF ALL EVIL IN OUR FAMILY.
SUPPOSE MONEY WERE NOT THE ROOT OF ALL EVIL IN OUR FAMILY
THEN I SUPPOSE WE WOULD BE SMOTHERED IN MOTHER LOVE.
YOU SAY THEN YOU SUPPOSE WE WOULD BE SMOTHERED IN MOTHER LOVE
NOT YOU AND I STUPID, JUST MY FAMILY AND ME
OH, I AND YOU STUPID
I SEE YOU RECOGNIZE THAT WORD. HAVE YOU HEARD IT A LOT?
PERHAPS IN YOUR FANTASY WE SEE EACH OTHER
SO, YOU ARE THAT KIND OF HEAD DOCTOR?
DOES IT PLEASE YOU TO BELIEVE I AM THAT KIND OF HEAD DOCTOR

(DEFPROP WORKER
(LAMBDA NIL
(PROG (SENTENCE SOBLIST KEystack)
(INC (INPUT TTY:X))
(TERPRI)
(TERPRI)
(INITIALIZE)
(READIN)
(ANALYZE)
(TERPRI)
(TERPRI)
(CLEANUP)
(GO A)))
EXPR)

(DEFPROP INITIALIZE
(LAMBDA NIL
(SETQ SOBLIST (APPEND OBLIST (SETQ SENTENCE (SETQ KEystack NIL))))
EXPR)

(DEFPROP READIN
(LAMBDA NIL
(PROG (WORD LETTER FLAG TERMINAL)
A (COND ((NULL (READWORD)) (GO B)))
(MAKESENTENCE)
(SETKEYSTACK)
B (BREAKANALYZE)
(COND ((NOT FLAG) (GO A)))
(SETQ SENTENCE (REVERSE SENTENCE))))
EXPR)

(DEFPROP ANALYZE
(LAMBDA NIL
(PROG (RULES PARSELIST DECOMP)
(SETQ KEystack
(APPEND KEystack
(LIST
(GET (QUOTE NONE)
(COND ((ZEROP
(SETQ FLIPFLOP
(PLUS 2. (MINUS FLIPFLOP))))
(QUOTE MEM))
(QUOTE LASTRESORT))))))
A (SETQ RULES (GET (CAR KEystack) (QUOTE RULES)))
B (SETQ DECOMP
(CAAR
(COND ((ATOM (CAR RULES))
(SETQ RULES (GET (CAR RULES) (QUOTE RULES)))
(RULES)))
(SETQ PARSELIST NIL)
(COND ((NOT (TEST DECOMP SENTENCE)) (SETQ RULES (CDR RULES)))
((AND (NOT (ATOM (CAR (SETQ RULES (CAR (ADVANCE))))))
(NOT (EQ (CAAR RULES) (QUOTE PRE))))
(RETURN (SENTPRINT (RECONSTRUCT (CAR RULES))))
((NOT (ATOM (CAR RULES)))
(SETQ SENTENCE (RECONSTRUCT (CADAR RULES)))
(SETQ RULES (CDDAR RULES)))
((EQ (CAR RULES) (QUOTE NEWKEY))
(SETQ KEystack (CDR KEystack))
(GO A)))
(GO B)))
EXPR)

```


MACRO-10*

for AL/COM

JANUARY, 1970

MACRO-10 is the name applied to both the symbolic assembly program, and to the powerful and versatile assembly language used in programming AL/COM computers. MACRO-10 is a two-pass assembler, operating in a minimum of 6K core. It performs many useful and unique functions, making machine language programming easier, faster, and more efficient. The assembler processes source program statements by translating mnemonic operating codes to binary machine codes, relating symbols to numeric values, assigning relocatable or absolute core addresses for program instructions and data, and preparing an output listing of the program which notes any errors detected during assembly. MACRO also allows the programmer, by creating new language elements, to perform a series of specialized functions.

MACRO-10 Capabilities

Users of AL/COM Time-Sharing may write MACRO-10 programs and subroutines as a series of free-format statements. Here are some of the capabilities at the disposal of the MACRO programmer:

1. MACRO is extremely useful in writing subroutines to do special purpose bit manipulations on scalar and array variables for FORTRAN programs.
2. Powerful "macro" capabilities permit entire coding sequences to be generated with a single statement. In some programs they may be used repeatedly, changing only the real arguments.
3. Functions and subroutines, to be called from FORTRAN IV programs, can easily be written in MACRO-10.

MACRO-10 LANGUAGE STATEMENTS

MACRO-10 programs are prepared, using SEDIT, as a sequence of statements. Each statement is normally written on a single line. MACRO-10 statements are virtually format-free. There are four types of elements in a statement. These elements are identified by their order of appearance and by a delimiting character or space. Statements are written in the general form:

label: operator operand, operand; comments <carriage return>

The assembler interprets and processes these statements, generating one or more binary instructions or data words, or performing an assembly process. A statement must contain at least one of the elements, and may contain all four types. Some statements are written with only one operand; others may have many.

Labels

A label is the symbolic name, created by the programmer to identify a statement. If present, the label, or labels, are written first in a statement, and are terminated by a colon (:).

Operators

An operator may be one of the 366 mnemonic machine instruction codes (see System Reference Manual), a command to monitor, or a pseudo-operation code which directs assembly processing. Programmers may also devise pseudo-ops to extend the power of the assembly language. An operator may also be a "macro" name, which calls a user-defined macro instruction. Like pseudo-ops, macros direct assembly processing, but possess unique power to handle repetition and to extend and adapt the assembly language.

Operands

Operands are usually the symbolic addresses of data to be accessed during execution, or input data or arguments of a pseudo-op or macro instruction. In every case, interpretation of operands depends upon the statement operator. Operands are separated by commas, and terminated by a semicolon (;) or carriage return.

Continued on other side

* MACRO-10 is the assembly language of the Digital Equipment Corporation PDP-10 Central Processors used in AL/COM Dual AL-10 Systems.

AL/COM is a direct access Computer Time-Sharing Service of Applied Logic Corporation • 1 Palmer Square • Princeton, N.J. 08540 • 609-924-7800

Comments

Notes may be added to a statement following a semicolon. They do not affect assembly or execution, but are useful in program listing, for analysis or debugging.

Symbols

The programmer may create his own unique symbols for statement labels, operators, and operands, and also utilize the standard machine instruction code symbols to direct processing. A symbol consists of one to six characters from the set of 26 letters (A–Z), 10 digits (0–9) and 3 special characters (\$, %, .). A symbol used as a *label* is "defined", and can be referenced by an instruction, or data word anywhere in the program. Symbols used as *operators* are predefined, either by the programmer, or by the assembler if a mnemonic machine instruction code, and control program assembly or processing. Symbols used as *operands* may be symbolic references to defined labels where the arguments to be used are to be found, or the values of constants or character strings.

The assembler processes symbols in source program statements by referencing its symbol table, relating all defined symbols to their binary machine codes. Initially the symbol table contains mnemonic "op" codes of the machine instructions, monitor and I/O command mnemonics, and the assembler "pseud-op" codes. As the source program is processed, new symbols, defined in the program, and by MACRO statements for use by the source program, are added to the table.

Example: a Fortran-callable subroutine to do a character fetch.

```
00100          TITLE   IGCHAR
00110      ;
00120      ;          CALL IGCHAR (HOLLER, J, M, N)
00130      ;
00140      ;          WHERE 'HOLLER' IS A HOLLERITH VARIABLE (INPUT) AND 'J'
00150      ;          IS THE NUMBER OF THE CHARACTER WHICH THE USER WISHES
00160      ;          EXTRACTED (INPUT). (1<J<5). THE SPECIFIED CHARACTER
00170      ;          IS RETURNED AS A 1H VARIABLE IN ARGUMENT 'M', AND AS
00180      ;          A RIGHT JUSTIFIED INTEGER IN VARIABLE 'N' ('M' AND
00190      ;          'N' OUTPUT).
00200      ;
00210          ENTRY    IGCHAR
00220      ;
00230 PNT:      POINT    7,@0(16),6
00240          POINT    7,@0(16),13
00250          POINT    7,@0(16),20
00260          POINT    7,@0(16),27
00270          POINT    7,@0(16),34
00280      ;
00290 SPACE4: ASCII  /      /
00300      ;
00310 IGCHAR: Z          ;      SAVE AC 16
00320          MOVE      1,@1(16)      ;      GET CHAR POSITION 'J'
00330          SKIPLE    1              ;      IF NOT 0<J<6
00340          CAIL      1,6
00350          MOVEI     1,1              ;      SET J TO 1
00360          LDB        0,PNT-1(1)     ;      FETCH BYTE
00370          MOVEM      0,@3(16)       ;      STORE IN VARIABLE 'N'
00380          IOR        0,SPACE4       ;      OR IN TRAILING SPACES
00390          ROT        0,-7           ;      MAKE IT THE FIRST CHAR
00400          MOVEM      0,@2(16)       ;      STORE IN VARIABLE 'M'
00410          JRA        16,4(16)       ;      RETURN !
00420          END

EXIT
↑C

.
```


AL/COM COBOL



NOVEMBER, 1968

AL/COM-COBOL is a version of COMPACT COBOL, or, as defined by USASI, a minimum subset of standard COBOL.

The following is a description of the differences between AL/COM-COBOL and standard COBOL.

1. Overall language considerations.

Data-names and numeric literals may not exceed 15 characters. All names must be unique. (To use DDT the first 6 characters of names must be unique.)

The space and the period are the only valid punctuation characters.

Only the figurative constants SPACE and ZERO, and their plurals, are permitted.

Condition names and the special register TALLY may not be used.

Indexing is limited to one dimension only. Each statement must be ended by a period.

2. Processing capabilities.

The standard COBOL verbs are available with the following exceptions: COMPUTE and EXAMINE are not part of this subset of COBOL; only option 1 of the PERFORM verb is permitted; the false path (ELSE or OTHERWISE) may not be used on IF statements.

Input-output operations are handled in the standard manner. The following input-output devices are available to the AL/COM-COBOL programmer: card reader, card punch, magnetic tape, disc, printer, and teletype. If no device is mentioned, the ACCEPT and DISPLAY verb assume the user's teletype.

A maximum number of 17 input and output files per job may be used. At this time, records may be fixed length only, and a file may be accessed only sequentially. Random access processing (reading and writing according to a defined ACTUAL KEY) will be available shortly. In addition, a REREAD option will be available. This will allow rereading (perhaps according to a different format) the last record read.

More on other side

TYPE FILE.COB

FILE.COB 11/19/68 1632 1-1-126

```
00010 IDENTIFICATION DIVISION.
00020 PROGRAM-ID. SAMPLE COBOL.
00030 ENVIRONMENT DIVISION.
00040 SOURCE-COMPUTER. PDP-6.
00060 OBJECT-COMPUTER. PDP-6.
00070 FILE-CONTROL.
00080     SELECT INFILE ASSIGN TO DRM1.
00090     SELECT OUTFILE ASSIGN TO DRM2.
00100 DATA DIVISION.
00110 FILE SECTION.
00120 FD INFILE DATA RECORD IS READIN.
00130 01 READIN.
00140     02 FILLER PICTURE X(40).
00150 FD OUTFILE DATA RECORDS ARE RESULTS, TIME-AND-DATE,
00160 PRIN-TOT.
00170 01 RESULTS.
00180     02 ACCT-NO PICTURE 9(5).
00190     02 NAME PICTURE A(20).
00200     02 AMOUNT PICTURE $Z,ZZ9.99.
00210 01 TIME-AND-DATE.
00220     02 TIME-FOR-TODAY PICTURE X(10).
00230     02 TODAYS-DATE PICTURE X(16).
00240 01 PRIN-TOT.
00250     02 FILLER PICTURE X(10).
00280     02 TOT-ACT PICTURE $ZZZ,ZZ9.99.
00290 WORKING-STORAGE SECTION.
00300 77 TEN-10 PICTURE 99V9 VALUE 10.0.
00310 77 ACCUMULATOR PICTURE S9(6)V99 VALUE ZERO.
00320 77 COUNTER PICTURE S999 VALUE ZERO.
00330 01 READ-STORAGE.
00340     02 FILLER PICTURE X(8).
00350     02 FIELD-1 PICTURE 9(5).
00360     02 FIELD-2 PICTURE A(20).
00370     02 FIELD-3 PICTURE S999V99.
00380     02 FORM-FEED PICTURE XX.
00390 PROCEDURE DIVISION.
00400 START. OPEN INPUT INFILE. OPEN OUTPUT OUTFILE.
00410     TIME TIME-FOR-TODAY. DATE TODAYS-DATE.
00420     WRITE TIME-AND-DATE.
00430 READIT. READ INFILE AT END GO TO FINISH.
00440     MOVE READIN TO READ-STORAGE.
00450     IF FIELD-1 IS GREATER THAN 500 PERFORM CALC
00460     THRU EXOUT. NOTE ACCOUNT NUMBERS BELOW 501
00470     ARE NOT PROCESSED.
00480 GOTO. GO TO READIT.
00490 CALC. MULTIPLY FIELD-3 BY TEN-10 GIVING AMOUNT
00500     ROUNDED ON SIZE ERROR ALTER GOTO TO PROCEED
00510     TO ERCALC.
00520     MOVE FIELD-1 TO ACCT-NO.
00530     MOVE FIELD-2 TO NAME.
00540     WRITE RESULTS AFTER ADVANCING 1 LINE.
00550     ADD FIELD-3 TO ACCUMULATOR.
00560     ADD 1 TO COUNTER.
00570     IF COUNTER IS NOT EQUAL TO 100 GO TO EXOUT.
00580     MOVE ACCUMULATOR TO TOT-ACT.
00590     WRITE PRIN-TOT.
00600     MOVE ZEROES TO COUNTER, ACCUMULATOR.
00610 EXOUT. EXIT.
00620 ERCALC. STOP "SIZE ERROR IN LAST RECORD".
00630 FINISH. IF COUNTER IS LESS THAN 100 GO TO LAST.
00640     CLOSE INFILE. CLOSE OUTFILE. STOP RUN.
00650 LAST. IF COUNTER IS EQUAL TO 0 GO TO MOVIT.
00655     MOVE ACCUMULATOR TO TOT-ACT.
00660     WRITE PRIN-TOT.
00670 MOVIT. MOVE 100 TO COUNTER.
00680     GO TO FINISH.
00690 END START.
```


SNOBOL[®]

for AL/COM

MAY, 1969

SNOBOL is a STRING ORIENTED SYMBOLIC LANGUAGE especially designed for the processing of character strings. It provides a powerful means for searching through character strings in order to match patterns, rearrange strings, and form new strings. It is useful in linguistics, the construction of psychological models, cryptanalysis, theorem proving, music and constructing bibliographies.

SNOBOL is especially useful in producing "front ends" for sophisticated FORTRAN and MACRO programs. These front ends can be quickly written and provide an excellent interface between the programs and non-computer oriented users. This relieves the veteran programmer from the arduous task of either writing the interface in the mathematical language of his program or training users in the use of the inflexible format restrictions associated with these languages. SNOBOL also provides an expedient method for reformatting of data files by offering simple operations for reading in data, for creating files by changing or deleting certain string sections of existing files, or adding sections to files. SNOBOL has a simple statement format, string oriented input and output, editing aids, dynamic storage, subroutine compatibility with Macro and Fortran programs, and line by line compatibility with Macro statements.

SNOBOL is an easy language for both computer-oriented and non-computer-oriented personnel to learn. Because of the simple statement format and uncomplicated I/O commands, the inexperienced may write a useful SNOBOL program in as little as two to four hours.

SNOBOL TERMINOLOGY

1. **String** refers to a series of characters. When used in a SNOBOL program, a string must be enclosed in quotes.
2. **Length** of a string indicates the number of characters, including blanks, within a string.
3. **Null String** is a string of length zero.
4. **String Name** is a title assigned to a string of characters.
5. **Label** is a name given to a SNOBOL statement.

SOME SNOBOL OPERATIONS

1. **Replacement** is the operation of assigning.
2. **Deletion** assigns the null string to a string name.
3. **Concatenation** is the procedure used to link two or more strings to form a new string.
4. **Transfer Commands** are provided for conditional and unconditional transfer of execution to some statement.
5. **Pattern Matching** operation is available for the comparison of strings.
6. **Arithmetic Operations** may be performed on integer numbers.
7. **Arithmetic Relations** are used to determine conditions between two numeric strings.
8. **Comments** may be written by preceding the comment with a semicolon.

More on other side

.TYPE SNOBO.SNO,FCODE.DAT,A,B,C

SNOBO.SNO 4/30/69 1449 343-1-1

```
00100      .OPIN "FCODE.DAT"      ; OPEN FILE FCODE.DAT FOR INPUT
00120      .READ LINE /F(FINISH)  ; READ A RECORD OF FCODE
00122
00140      LINE *DEPT* "+" =
00160      NEWDE: MATCH = DEPT
00165
00175      ; OPEN A FILE FOR OUTPUT,THE FILE NAME WILL BE THE
00176      ; SAME AS THE CONTENTS OF THE STRING NAME MATCH
00180      .OPOUT MATCH
00195
00200      BEGIN: LINE *SOSEC* "+" =
00220      LINE *NAME* =
00240      NEWLI = DEPT " " SOSEC " " NAME
00260      .WRITE NEWLI
00262
00265      ; READ THE NEXT RECORD FROM INPUT FILE--IF AN END OF
00275      ; FILE HAS BEEN REACHED TRANSFER COMMAND TO FINISH
00280      .READ LINE /F(FINISH)
00290
00300      LINE *DEPT* "+" =
00320      DEPT MATCH /S(BEGIN)
00330      ; CLOSE THE CURRENT OUTPUT FILE AND TRANSFER CONTROL
00331      ; TO NEWDE
00340      .CLOUT /(NEWDE)
00360      FINISH: .END ; END OF THE PROGRAM
```

Search LINE until a "+" character is encountered. Assign all characters preceding that "+" to DEPT. Delete those characters and "+" from LINE.

Replace the contents of MATCH by the contents of DEPT.

Search LINE for another "+" character. Assign all characters preceding that "+" to SOSEC and delete those characters and "+" from LINE.

Assign the remaining characters in LINE to NAME and delete those characters from LINE.

Replace the contents of NEWLI by the contents of DEPT, a space, SOSEC, a space, and NAME.

Write the contents of NEWLI on the output file.

Compare the contents of DEPT to the contents of MATCH. If the comparison is successful, transfer execution to the statement labelled BEGIN.

FCODE.DAT 4/30/69 1449 343-1-1

```
00100      A+111-11-1111+JONES,JOHN
00120      A+112-21-2222+SMITH,HAROLD
00140      A+345-92-6178+MURRAY,ROBERT
00160      A+764-42-8196+BROWN,MARY
00180      B+248-13-7649+WARGA,BETH
00200      B+114-41-1441+SALERNO,CHERYL
00220      B+121-76-3289+BROADWAY,FLORENCE
00240      C+139-47-8291+BAKER,JOHN
00260      C+281-42-7639+BLIVEN,LAUREEN
00280      C+276-54-3210+JOHNSON,LOIS
00300      C+468-32-5986+SCHWARTZ,LINDA
00320      C+543-12-9876+COLTON,EDITH
```

A 4/30/69 1449 343-1-1

```
A 111-11-1111 JONES,JOHN
A 112-21-2222 SMITH,HAROLD
A 345-92-6178 MURRAY,ROBERT
A 764-42-8196 BROWN,MARY
```

B 4/30/69 1449 343-1-1

```
B 248-13-7649 WARGA,BETH
B 114-41-1441 SALERNO,CHERYL
B 121-76-3289 BROADWAY,FLORENCE
```

C 4/30/69 1449 343-1-1

```
C 139-47-8291 BAKER,JOHN
C 281-42-7639 BLIVEN,LAUREEN
C 276-54-3210 JOHNSON,LOIS
C 468-32-5986 SCHWARTZ,LINDA
C 543-12-9876 COLTON,EDITH
```


AL/COM's SEDIT, or Super EDITor, provides our users with a remarkable facility to create and edit sequenced files in ways highly sophisticated yet simple for any user to comprehend. SEDIT's features include:

1. Automatic Indexing—

In "create" mode the user can command SEDIT to type out 5-digit sequence numbers of lines being inserted.

2. Short Cut Commands—

The **Replace** command allows the user to replace a character string by another of any size, or by nothing. Deleting and reinserting the entire line is unnecessary.

To insert or replace one line, simply type the sequence number, a tab, and then the line.

3. Powerful String Editing As Well As Line Editing Capabilities With These Commands—

Append	Multiple Look with optional Replace
Build into the/BUFFER	Multiple Replace
Delete	Multiple Search
Hack and Replace	Replace
Insert	Search
Look and Append	Type or Print
Look and Delete	Update

4. Convenient Switches To Speed Up Editing—

/NOASTERISK and /NOINNUMBER suppress most program responses, allowing the user to type commands and insert material at his own speed.

With switches, the user can define his own "escape" key, or incorporate the changes he has made so far into the permanent version of his file.

5. Automatic Saving Of File On Disc At End Of The Edit

Two experimental features presently under consideration for SEDIT are:

1. File Update Feature—

The user creates a file containing SEDIT commands. He then instructs SEDIT to edit his program from this file of commands. All corrections are made without user sitting at TTY. The File Update feature offers real and computer time savings.

2. Buffer Feature—

Any segment of a created program can be stored in SEDIT's auxiliary buffer, then reinserted in the main program or in subroutines having these lines in common.

More on other side

↑C

Sample Creation of File Named TEST . FOR

R SEDIT

*ETEST.FOR\$*I100

```
00100      TYPE 500
00110      10    TYPE 600
00120      ACCEPT 700,NFIRST,ILAST,IFIRST
00130      30    DO 100 I=IFIRST,ILAST
00140      J=IPRIME(I)
00150      GO TO (50,100),J
00160      50    TYPE 800
00170      100   CONTINUE
00180      GO TO 10
00190      500   FORMAT(///)
00200      600   FORMAT(IX,'TYPE NFIRST,ILAST')
00210      700   FORMAT(2I)
00220      800   FORMAT(I8,<\,' IS A PRIME')
00230      END
00240      $*E
```

"\$*E" where E signals end of file and also saves file automatically.

Multiple Search from start (1) to end of file (Z) for every occurrence of "J".

Look at I#160 and Append to it "I".

EXIT

↑C

Editing TEST . FOR

.R SEDIT

*ETEST.FOR

*90 C- TO INSERT OR OVERWRITE A LINE TYPE LINE#,TAB,LINE OF TEXT

*MS1,Z\$J\$

```
00140      J=IPRIME(I)
00150      GO TO (50,100),J
```

SEARCH ENDS*LA160

00160 50 TYPE 800,I

*T.

00160 50 TYPE 800,I

*HR120\$,IF\$\$*MRL1,Z\$NFIRST\$IFIRST\$

00120 ACCEPT 700,NFIRST,ILAST

?Y

00200 600 FORMAT(IX,'TYPE NFIRST,ILAST')

?Y

SEARCH ENDS*E

EXIT

↑C

Corrected File, TEST . FOR

.TYPE TEST.FOR

TEST.FOR 12/26/68 1410 1-1-124

00090 C- TO INSERT OR OVERWRITE A LINE TYPE LINE#,TAB,LINE OF TEXT

```
00100      TYPE 500
00110      10    TYPE 600
00120      ACCEPT 700,IFIRST,ILAST
00130      30    DO 100 I=IFIRST,ILAST
00140      J=IPRIME(I)
00150      GO TO (50,100),J
00160      50    TYPE 800,I
00170      100   CONTINUE
00180      GO TO 10
00190      500   FORMAT(///)
00200      600   FORMAT(IX,'TYPE IFIRST,ILAST')
00210      700   FORMAT(2I)
00220      800   FORMAT(I8,' IS A PRIME')
00230      END
```

EXIT

↑C

Note: "\$" indicate "Escape" or "Altmode".

Type last line examined by SEDIT ("." = last line).

Multiple-Replace-Look searches for all occurrences of NFIRST, types line, and responds with "?" User responded Y-return, i.e. yes, replace, and continue search.

Replace "IF" by nothing and hack the rest of the line from there.

RSMPLX

for AL/COM

(RSMPLX Revised Simplex Subroutine) FEBRUARY, 1970

RSMPLX is a group of eleven Fortran-coded subroutines, which together constitute a powerful mathematical tool for solving linear programming problems. Although of relatively recent origin, linear programming finds application in a wide range of disciplines such as management science, game theory, industrial production, planning and econometric modeling.

The basic technique of linear programming minimizes or maximizes a linear function of several variables subject to certain imposed linear constraints. In industrial management, for example, the function maximized may represent profit while the constraints relate to the availability of labor, time and raw materials.

In the mathematical statement of the linear programming problem, the function to be minimized is defined in terms of N variables, X_i , and N cost coefficients, C_i , where $i = 1 \dots N$ as:

$$\sum_{i=1}^N C_i X_i \quad (\text{the object or cost function})$$

The variables X_i are subject to M constraint equations (where $M < N$) of the form:

$$\sum_{i=1}^N A_{ij} X_i = B_j \quad (j = 1 \dots M)$$

In the constraint equations, A is known as the "coefficient matrix", and B as the "right hand side". A further limitation imposed upon the solution is that all of the variables, $X_i > 0$.

In general, the cost function is minimized when $N-M$ of the variables are set to zero, and the remainder are greater than zero. RSMPLX obtains the solution by the well-known revised simplex algorithm.

Features of AL/COM RSMPLX

1. The user need call only the primary RSMPLX subroutine. Calls to other subroutines within the linear programming package are made from RSMPLX.
2. This subroutine package is free of input/output statements permitting it to be incorporated in other programs.
3. RSMPLX allows the user to generate data in storage, call the subroutine to process the data, and store the results for later output or additional future processing.
4. There is no restriction on the signs of the B_j 's.
5. The program may be restarted with artificial, partial or full basis.
6. Parameters are passed to RSMPLX via COMMON blocks.
7. Problems up to a maximum of approximately 100 rows by 150 columns, or the equivalent, may be solved. Arrays are dimensioned in the calling program, simplifying setup and changes.
8. Provision is made for use of multiple objective functions with the same coefficient matrix. Multiple right hand sides are also possible. Constraints may be imposed or removed readily.
9. Solutions to the "dual" problem can be readily obtained.
10. A number of other options are available, including "mixed pricing" or actual objective.

Continued on Reverse Side

EXAMPLE:

A very simple example in two variables with three constraints is presented here to show the input, iterations, and initial and final tableaus of RSMPLX.

The file LPMAIN.FOR shows production line data for an imaginary distillery producing two blended products, x and y. Profit on x is \$1.50, while y yields \$1.25 profit.

Constraints are:

1. Pot still capacity = 5000 oz. per day. Blend x requires 20 oz. per bottle; blend y, 10 oz. per bottle (1/5 gal.).
 $20x + 10y \leq 5000$
2. Filter and blending equipment can process 500 fifths per day. Blend x is filtered once, while blend y receives double filtration.
 $x + 2y \leq 500$
3. Bottler requires three operations on blend x bottle, while only two are needed on blend y bottle. Capacity is 800 operations per day.
 $3x + 2y \leq 800$
4. Objective function (expressing maximum profit) is:
maximize $M = 1.5x + 1.2y$

The final tableau reveals that $X=150$, $Y=175$ (5ths per day). The significance of 250 (line 3, col. 1, solution vectors) is that full capacity of still was not used. It could have produced an additional 250 oz. The profit on the two products is: $1.5 \times 150 = \$225$ plus $1.2 \times 175 = \$210 = \435 (maximum).

```
LPMAIN.FOR PAGE 1

00100      BLOCK DATA
00110      COMMON /A/A(20) /B/B(4) /X/X(4) /JH/JH(4)
00120      COMMON /E/E(16) /P/P(4) /Y/Y(4) /KB/KB(5)
00130      COMMON /OUTCON/KOUT(8) /ERROR/TERR(8)
00140      COMMON /INCON/INFLAG,N,ME,M,MF,MC,NCUT,NVER
00150      COMMON /TOL/ TPIV,TZERO,TCOST,TECOL,PRM
00160      END
00170
00180      COMMON /A/A(1) /B/B(1) /X/X(1) /JH/JH(1)
00190      COMMON /E/E(1) /P/P(1) /Y/Y(1) /KB/KB(1)
00200      COMMON /OUTCON/KOUT(1) /ERROR/TERR(1)
00210      COMMON /INCON/INFLAG,N,ME,M,MF,MC,NCUT,NVER
00220      COMMON /TOL/ TPIV,TZERO,TCOST,TECOL,PRM
00230      EQUIVALENCE (KOUT(1),K),(KOUT(2),ITER),(KOUT(3),INVC),
00240      1 (KOUT(4),NUMVR),(KOUT(5),NUMPV),(KOUT(6),INFS),
00250      2 (KOUT(7),JM)
00260      INFLAG=4
00270      TYPE 200
00280      ACCEPT 210,ME,N
00290      TYPE 220
00300      ACCEPT 210,MF,M
00310      TYPE 230
00320      ACCEPT 210,MC
00330      TYPE 240
00340      ACCEPT 210,NCUT,NVER
00350      TYPE 250
00360      ACCEPT 260,TPIV,TZERO,TCOST,TECOL
00370      DO 100 I=1,ME
00380      TYPE 270,I
00390      100 ACCEPT 260,(A(II),II=I,ME*N,ME),B(I)
00400      CALL RSMPLX
00410      TYPE 280,K
00420      DO 110 I=1,N
00430      XX=0.0
00440      PP=0.0
00450      J=KB(I)
00460      IF(J.EQ.0)GO TO 110
00470      PP=P(J)
00480      XX=X(J)
00490      110 TYPE 290,I,XX,PP
00500      TYPE 300,X(1)
00510      STOP
00520      200 FORMAT('ENTER PROBLEM: '/' ROWS, COLS IN "A"...? ')
00530      210 FORMAT(2I)
00540      220 FORMAT('FIRST, LAST CONSTR. IN "A"...? ')
00550      230 FORMAT('ROW # OF OBJ FCN...? ')
00560      240 FORMAT('MAX ITERS & REINVS...? ')
00570      250 FORMAT('TOLERANCES( PIV,X,COST & E)...? ')
00580      260 FORMAT(10F)
00590      270 FORMAT('ROW',I1,'...? ')
00600      280 FORMAT(' OUTPUT CONDITION: ',I2,'/' SOLUTION VECTORS:')
00610      290 FORMAT(13,2F)
00620      300 FORMAT('OBJECTIVE VALUE: ',F,/)
00630      END
```

```
.EXE LPMAIN.FOR PUB:RSMPLX
COMPILING: LPMAIN.FOR
LOADING.
```

```
CORE 4.
START 000547
```

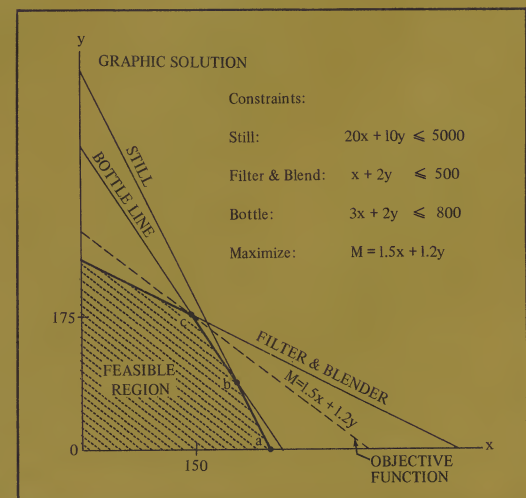
```
ENTER PROBLEM:
ROWS, COLS IN "A"...? 4,5
FIRST, LAST CONSTR. IN "A"...? 2,4
ROW # OF OBJ FCN...? 1
MAX ITERS & REINVS...? 10
TOLERANCES
( PIV,X,COST & E)...? .001,.001,-.001,-.001
ROW1...? -1.5 -1.2
ROW2...? 20 10 1 0 0 5000
ROW3...? 1 2 0 1 0 500
ROW4...? 3 2 0 0 1 800
```

OUTPUT CONDITION: 1

```
SOLUTION VECTORS:
1 150.0000000 0.0000000
2 175.0000000 0.4500000
3 250.0000000 0.1500000
4 0.0000000 0.0000000
5 0.0000000 0.0000000
```

OBJECTIVE VALUE: 435.0000000

```
EXIT
↑C
.
```



MATHEMATICAL SUBROUTINES



(Mathematical Subroutines Library) DECEMBER, 1970

This set of computation subroutines, for use in FORTRAN IV programming, is intended to aid the user in the development of his own FORTRAN programs. These subroutines are arranged for indexing purposes into five groups: statistics, mathematics, polynomials, calculus, and special functions. They can be applied to the solution of many problems in industry, science and engineering.

Some of the characteristics of AL/COM's MS are:

1. Over 265 subroutines are presented. Both single and double precision modes are available.
2. All the subroutines are free of input/output statements.
3. Subroutines do not contain fixed maximum dimensions for the data arrays named in their calling sequences.
4. Many matrix manipulation subroutines handle symmetric and diagonal matrices (stored in economical, compressed formats) as well as general matrices. This can result in considerable saving in data storage for large arrays.
5. The use of the more complex subroutines (or groups of them) is illustrated in the program manual by sample main programs with input/output.
6. All Mathematical Subroutines are in relocatable (binary) form, ready to be loaded with your program, saving compilation cost and time. They are stored in library files on device PUB:.
7. Related mathematic subroutines are stored in a single class file within the library. For example: all polynomial routines are stored in POLY.LIB
8. Documentation on every subroutine is available. The files are conveniently arranged in a tree structure to facilitate retrieval, and may be typed with the command:

EXPLAIN <subroutine name>.

Documentation of Subroutines

The following example is illustrative of this tree-structured documentation.

User wishes to add, multiply, and invert matrices:

EXPLAIN MATH (See above.)

System responds with description of classification method, list of classes, and instructions.

User selects the following classification:

MAT, DMAT—Matrices: Storage, operations, inversion, systems of linear equations, and eigenanalysis.
(prefix "D" indicates double precision)

He then selects subset "MAT" and types:

EXPLAIN MAT <return>

... which types out a description of all matrix manipulation routines in the library. User selects subroutines MADD, MPRD and MINV2 (which respectively add, multiply, and invert matrices) as suitable. He next types:

EXPLAIN MADD MPRD MINV2 <return>

... which types out complete descriptions and operating instructions for all 3 routines. He is now ready to use these routines.

Use of Subroutines

A FORTRAN listing of many of the Mathematical Subroutines, and a detailed explanation of their use, is available from Applied Logic Corporation or its Associates.

Continued on other side

This simplified example illustrates the commands necessary to CALL a mathematical subroutine from the main program and to RETURN control to the main program:

A file named SIGMA.FOR containing the main program the user has created would appear as follows:

```
...  
...  
...  
N=10  
M=20  
CALL MADD (AR1, AR2, AR3, N,M)  
...  
CALL MPRD (A,B,R,N,M,MSA,MSB,L)  
...  
CALL MINV2 (A,N,D,L,M)  
...
```

The subroutines called by SIGMA.FOR are located in the MAT group. To use these subroutines, type:

EXECUTE SIGMA.FOR PUB:MAT.LIB <return>

THE AL/COM MATHEMATICAL SUBROUTINES LIBRARY IS MAINTAINED BY CLASSES OF SUBROUTINES. THE INDIVIDUAL SUBROUTINES BELONGING TO A SINGLE CLASS WITH NAME "XXXX" ARE DOCUMENTED IN THE TWO FILES XXXX AND DXXXX. THE LATTER FILE IS FOR DOUBLE PRECISION VERSIONS OF THE ROUTINES. COMPLETE DOCUMENTATION OF EACH SUBROUTINE WITHIN A CLASS CAN BE LISTED TO THE TELETYPE, TYPE:

EXPLAIN <SUBROUTINE NAME> <RETURN>

ALL SUBROUTINES IN THE PACKAGE ARE AVAILABLE ALREADY COMPILED AND IN RELOCATABLE LIBRARY FORMAT. TO USE ROUTINES FROM CLASS XXXX, TYPE:

EXECUTE <PROGRAM NAME> PUB:XXXX.LIB

FROM TIME TO TIME NEW ROUTINES WILL BE ADDED TO THE PACKAGE BUT THE BASIC TREE STRUCTURE FOR DOCUMENTATION WILL REMAIN UNCHANGED.

CLASSES:

STAT1,DSTAT1--STATISTICS: DATA SCREENING, DESIGN ANALYSIS, DISCRIMINANT ANALYSIS, FACTOR ANALYSIS, AND TIME SERIES.

STAT2,DSTAT2--STATISTICS: CORRELATION AND REGRESSION, NONPARAMETRIC STATISTICS, GENERATION OF RANDOM VARIATES, DISTRIBUTION FUNCTIONS, ELEMENTARY STATISTICS, AND MISCELLANY.

MAT,DMAT--MATRICES: STORAGE, OPERATIONS, INVERSION, SYSTEMS OF LINEAR EQUATIONS, AND EIGENANALYSIS.

POLY,DPOLY--POLYNOMIALS: OPERATIONS, ROOTS, AND SPECIAL TYPES.

CALC1,DCALC1--CALCULUS: ROOTS OF NONLINEAR EQUATIONS, EXTREMUM OF FUNCTIONS, PERMUTATIONS, SEQUENCES, INTERPOLATION, APPROXIMATION, AND SMOOTHING.

CALC2,DCALC2--CALCULUS: NUMERICAL QUADRATURE, NUMERICAL DIFFERENTIATION AND ORDINARY DIFFERENTIAL EQUATIONS.

SPEC,DSPEC--SPECIAL FUNCTIONS.

CPM for AL/COM

(Critical Path Method) FEBRUARY, 1970

CPM (Critical Path Method) scheduling is an interactive program for solving the general problem of identifying those steps in a complex project which are critical with regard to time, and quantifying their contributions to potential delay in completion of the project.

Critical path problems arise whenever the number and interrelationship of the elementary steps prohibits an ordinary "common sense" solution. CPM answers the question: "when should each individual job be begun, and when must it be completed to keep project time or costs at a minimum?" CPM is useful to management and project leaders in a number of ways. It is an aid in formulating plans and developing realistic goals, since the effect of alternative decisions upon schedules and costs can be tested before implementation.

It is a proven method for focusing attention upon those tasks demanding priority handling, and in helping to define the best "trade-offs" in costs, resources or time, to meet project deadlines. CPM provides periodic progress reports identifying potential problems before they become crucial.

PHASE 1: DATA ENTRY

Data for input into AL/COM CPMNET program is entered into a user's file. Data includes network nodes and time estimates for each activity. Descriptions of each activity may also be included. The data file is usually created by following a previously prepared "arrow diagram" depicting the relationship of the various activities, and giving a time estimate for each activity.

PHASE 2: PROJECT SCHEDULE CALCULATIONS

The initial or network scheduling phase of the program calculates, for each activity, the earliest possible start day and the earliest possible finish day. It also calculates the last day an activity can begin and still finish without delaying other activities in the network. The numerical difference between early and late start days is the "total float" which indicates the "degree of criticality" of that activity. All activities lying on the "critical path" will have a total float time of 0, since "early start" and "late start" days will then coincide.

PHASE 3: PROJECT CONTROL AND MONITORING

Control and monitoring of the project is facilitated by having on hand listings of network activities that have been sorted in certain desirable ways. Therefore, AL/COM has built into its CPM program a sorting routine which will sort the network activities in any desired order. For example: a sort by "total float" will list activities from most critical to least critical, or for an on-going project, a sort by "early finish times" will place emphasis on lack of progress.

PHASE 4: CALENDAR DATING AND REPORTING

The final section of the program contains the option of assigning calendar dates to relative scheduling days. Provision is made for either 5 or 6 work days per week, and holidays and special non-work days may be entered. Reports may be generated in a variety of formats depending on the desired content.

COST ANALYSIS

Cost analysis is useful when normal project duration, determined in the scheduling phase, must be shortened. Knowing the "total float" or "degree of criticality" of each activity singles out those activities that must be crashed in order to shorten over-all project duration.

Continued on other side

EXAMPLE:

.TYPE TEST.CPM

TEST.CPM 1-27-70.943

```
00100 10,20,1,REVIEW EXISTING MAPPING
00110 20,30,2,DETERMINE FLIGHT PLAN
00120 30,40,4,FIELD LAY-OUT OF TARGETS
00130 40,50,3,ESTABLISH HORIZONTAL CONTROLS
00140 40,60,1,FLY FLIGHT
00150 50,70,0,DUMMY
00160 60,70,5,MAKE DIA-POSITIVES
00170 60,80,5,DEVELOP PHOTOS
00180 70,90,3,COMPILE FIRST 1/2 HARDCOPY
00190 80,100,3,COMPOSE MOSAIC
00200 90,100,0,DUMMY
00210 90,120,3,FINISH COMPILING HARDCOPY
00220 100,110,5,LAY CL FIRST 1/2 HARDCOPY
00230 110,120,0,DUMMY
00240 110,130,2,STRIP CL PROFILE 1/2
00250 120,130,5,FINISH LAYING CENTERLINE
00260 130,140,2,FINISH STRIPPING PROFILE
00270 140,150,1,SHIFT CL DUE TO PROFILE
00280 150,160,1,REVISED PROFILE INFORM
```

AL/COM CPM PROGRAM IS READY 09:45 27-JAN-70

INPUT FILE NAME: TEST.CPM

PROJECT ID: TESTING FOR INFORMATION

PHASE I STARTS !

NO. OF ACTIVITIES = 19

PHASE II STARTS !
NO DETECTABLE ERRORS !

INDICATE TYPE OF SORT: CR

REQUESTED SORT: CRITICALITY

INDICATE REPORT FORMAT: ES

REQUESTED FORMAT: EARLY START

ENTER OUTPUT FILE NAME: (OR TTY:) TTY:

REPORT TO INCLUDE ACTIVITY DESCRIPTIONS (YES OR NO) ? YES

CALENDAR DATING (YES OR NO) ? YES

5 OR 6 DAY WORK WEEK ? 5

PROJECT STARTING DATE ? 1,1,70

TESTING FOR INFORMATION

PAGE 1

I	J	DUR	ACTIVITY DESCRIPTION	EARLY START	TOTAL FLOAT
10	20	1	REVIEW EXISTING MAPPING	2JAN70	0
20	30	2	DETERMINE FLIGHT PLAN	5JAN70	0
30	40	4	FIELD LAY-OUT OF TARGETS	7JAN70	0
40	60	1	FLY FLIGHT	13JAN70	0
60	70	5	MAKE DIA-POSITIVES	14JAN70	0
60	80	5	DEVELOP PHOTOS	14JAN70	0
70	90	3	COMPILE FIRST 1/2 HARDCOPY	21JAN70	0
80	100	3	COMPOSE MOSAIC	21JAN70	0
90	100	0	DUMMY	26JAN70	0
100	110	5	LAY CL FIRST 1/2 HARDCOPY	26JAN70	0
110	120	0	DUMMY	2FEB70	0
120	130	5	FINISH LAYING CENTERLINE	2FEB70	0
130	140	2	FINISH STRIPPING PROFILE	9FEB70	0
140	150	1	SHIFT CL DUE TO PROFILE	11FEB70	0
150	160	1	REVISED PROFILE INFORM	12FEB70	0
90	120	3	FINISH COMPILING HARDCOPY	26JAN70	2
40	50	3	ESTABLISH HORIZONTAL CONTROLS	13JAN70	3
50	70	0	DUMMY	16JAN70	3
110	130	2	STRIP CL PROFILE 1/2	2FEB70	3